# Project 1: Model Selection via Cross Validation and Regularization, Model Evaluation, and (Generalized) Linear Regression; Application: Polynomial Curve-Fitting Regression for Deficit Data

## Overview

In this project you will apply polynomial curve-fitting for regression on a data set of 1939-2000 Federal Budget Deficit Data. [1] You will use *cross-validation (CV)* for model selection. You will use CV to select the optimal value for the degree of a low-degree polynomial to use on all the training data. You will also use CV to find the optimal regularization parameter to use to learn the best, in terms of *regularized squared-error regression*, polynomial of degree no larger than 12 that can be inferred from the training data. You will use a *test set* to evaluate the *root-mean-squared-error (RMSE)* of the optimal polynomial models you selected for learning.

Specifically, recall that the RMSE that a hypothesis $h$ achieves on the *hold-out dataset*

$$D_{\text{ho}} = \left\{ \left( x_{\text{ho}}^{(1)}, y_{\text{ho}}^{(1)} \right), \ldots, \left( x_{\text{ho}}^{(m_{\text{ho}})}, y_{\text{ho}}^{(m_{\text{ho}})} \right) \right\}$$

of $m_{\text{ho}}$ examples during each fold of CV is given by

$$\text{RMSE}(h; D_{\text{ho}}) \equiv \sqrt{\frac{1}{m_{\text{ho}}} \sum_{l=1}^{m_{\text{ho}}} \left( y_{\text{ho}}^{(l)} - h\left( x_{\text{ho}}^{(l)} \right) \right)^2} .$$

As stated in the previous paragraph, in this project, the hypothesis function $h$ would correspond to some non-regularized and regularized polynomial from the different hypothesis classes under consideration. You will need to *compute and record the RMSE* for *each* $h$ found for *each* corresponding hypothesis class on *each* fold of CV. Then you will need to *compute the average RMSE* that each hypotheses $h$'s, for *each* hypothesis class, achieved *over the different folds.* For each case of the low-degree non-regularized and the regularized 12-degree polynomial, you will *select the hypothesis-class model achieving the minimum average RMSE.* Such a hypothesis class corresponds to the *best CV model class* for your final hypothesis in *each* case. Finally, you will *apply the ML curve-fitting algorithm* on *all* the training data, using the best CV model class you found for *each* case.

### Deficit Data

You are given a dataset of some of the U.S.A. federal budget deficit (in billions of dollars) in current (i.e., as of 2000) dollars for a number of years between 1939 and 2000, not necessarily consecutive. The only input feature is the year. The output is the deficit amount. The files `deficit_train.dat`

[1] `http://www.oswego.edu/~kane/econometrics/deficit.htm`

and `deficit_test.dat` contain the training and test datasets, respectively. Each consists of two columns. The first column corresponds to the calendar-year values (*input*) and the second column are the deficit values (*output*). (**NOTE:** The examples are *not* chronologically sorted by the year input.)

For computational/numerical convenience, you must "normalize" the input values via a simple linear transformation that leads to the average of the input values being 0 and the (empirical, unbiased estimator of the) standard deviation being 1. Similarly, linearly transform the output so that the average of the output values is 0 and the (empirical, unbiased estimator of the) standard deviation is 1. Because the data transformation step vu normalization is a part of the learning algorithm, when learning using each CV training data for each fold, the model must remember the normalization parameters computed using the CV training data for that fold so the model can use that tranformation to normalize new inputs $x$ (i.e., new calendar years) into $\widetilde{x}$ to obtain an estimate output value $\widetilde{y}$ from the learned model in the transformed/normalized space that can then be used to recover the output $y$ in the original non-transformed/unnormalized space normalization (i.e., the estimated unnormalized deficit value $y$ from the new calendar year $x$), as described and discussed during lecture. Those are the transformed input-output pairs you will need to estimate the *training and test RMSE* of the regression hypotheses you learned after CV model selection.

## Regularized Squared-Error Regression using Polynomial Hypothesis Classes

For the deficit data, we have a single real-valued input feature (the normalized year) and a single real-valued output (the deficit amount), so that the domain, or feature space, and the range, or output space, are both one-dimensional. Note that, for numerical reasons, the input-output example pairs must be appropriately normalized, as described above.

In this project, you will use polynomials up to degree $d$ as the hypothesis class. Hence, recall that each hypothesis $h$ in our class has the form $h(x) = \sum_{i=0}^{d} w_i x^i = w_0 + w_1 x + w_2 x^2 + \ldots + w_d x^d$. Also recall that if the training data is of size $m$, and we denote the $l$th example as $\left(x^{(l)}, y^{(l)}\right)$, the $(\ell_2)$-regularized squared-error function is

$$L(\mathbf{w}) = \frac{1}{2} \sum_{l=1}^{m} \left( y^{(l)} - \sum_{i=0}^{d} w_i \left( x^{(l)} \right)^i \right)^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

where $\|\mathbf{w}\|_2 \equiv \sqrt{w_0^2 + w_1^2 + \cdots + w_d^2} = \sqrt{\sum_{i=0}^{d} w_i^2}$ is the $\ell_2$-norm, or magnitude, of the vector of polynomial coefficient-weights.

Recall that the idea of adding the regularization term is to penalize for large-magnitude coefficient-weights which tend to lead to more complex polynomial curves, and thus are more likely to overfit the training data. Note that when $\lambda = 0$, we have the standard squared-error function used for polynomial curve-fitting regression, without regularization. Thus, if we do not use regularization ($\lambda = 0$), minimizing $L(\mathbf{w})$ is equivalent to minimizing RMSE.

In this project, you will consider several values for the degree of the polynomial $d = 0, 1, \ldots, 12$ without regularization ($\lambda = 0$), and for $d = 12$, consider several values for the regularized parameter $\lambda = 0, \exp(-25), \exp(-20), \exp(-14), \exp(-7), \exp(-3), 1, \exp(3), \exp(7)$, where $\exp(z) = e^z$ is the natural-exponential function.

### The ML Regression Algorithm

In this project you are asked to apply the learning algorithm discussed in class, including the data normalization step stated above to deficit training data and to appropriate evaluate the learned classifier on deficit test data. You can either implement the entire algorithm from scratch or write code that uses appropriate existing libraries to perform the learning and evaluation tasks. Note that if you use existing libraries, you must make sure to use the appropriate methods and call them with the appropriate parameters to produce *exactly the same results that you would have obtained if you had coded the entire process yourself*; that is, you must understand exactly how the methods in the libraries you use work and what they are producing when you call them.

## Learning to Predict U.S.A. Federal Budget Deficit Amounts

You will learn the best $d$-degree polynomial using 6-fold CV on the training data to select the optimal polynomial-degree value $d$ to use from the set $\{0, 1, 2, 3, 4, 5, \ldots, 12\}$ (that is, $d = 0$ means using just a constant, $d = 1$ means standard linear regression, etc., up to $d = 12$ degree polynomial).

For CV, you must create each fold by splitting the examples in the training dataset using the *same order* as they appear in the data file. Specifically, the *first* fold corresponds to the examples with indexes $1, 2, 3, 4, 5, 6, 7$; the *sixth* fold corresponds to the examples with indexes $36, 37, 38, 39, 40, 41, 42$; and similarly for the indexes to examples in each of the other folds.

You will also learn the regularized coefficient-weights of a 12-degree polynomial using 6-fold CV on the training data to select the optimal regularization parameter $\lambda$ to use from the set $\{0, \exp(-25), \exp(-20), \exp(-14), \exp(-7), \exp(-3), 1, \exp(3), \exp(7)\}$.

During each of the 6 folds of CV, you will obtain a RMSE value for each $d$ or $\lambda$ value considered. Record those values so that you can evaluate/report the *average* of the RMSE values you obtained for each $d$ or $\lambda$ value during each of the 6 folds of CV.

Suppose $d^*$ or $\lambda^*$ is the $d$ or $\lambda$ value with the lowest RMSE after 6-fold CV. Obtain the coefficient-weights of the $d^*$-degree polynomial using *all* the training data. Also run regularized polynomial curve-fitting regression for a 12-degree polynomial using *all* the training data with $\lambda^*$ as the regularization parameter. Report the *coefficient-weights* for $d^*$ and $\lambda^*$, and the corresponding values of the *training and test* RMSE obtained for the resulting polynomials. For *extra credit*, plot all the training data along with the resulting polynomial curves for $d^*$ and $\lambda^*$, for the range of years 1935-2005 as input.

Once again, as with the ML regression algorithm described abov, you can either code the CV process, and the final learning and evaluation tasks from scratch or use existing libraries or tools as long as you call those libraries or set up those tools in a way that would produce the same results as you would have obtained if you had coded the entire process yourself.

## What to Turn In

You must submit the following (electronically via Canvas):

1. A **written report** (*in PDF*) that includes (1) the averages of the RMSE values you obtained during the 6-fold CV for each case; (2) the optimal degree $d^*$ and regularization parameter $\lambda^*$ you obtained via the 6-fold CV; (3) the coefficient-weights of the $d^*$-degree polynomial and the regularized 12-degree learned using $\lambda^*$ on all the training data; (4) the training and test

RMSE of the final, learned polynomials; (5) the 2 plots containing all the training data along with the resulting polynomial curves for $d^*$ and $\lambda^*$, for the range of years 1935-2005 as input; and (6) a brief discussion of your findings and observations.

2. All your **code and executable** (as a tared-and-gziped compressed file), with instructions on how to run your program. A platform-independent standalone executable is preferred; otherwise, also provide instructions on how to compile your program. In general, the submitted program must be able to be executed as a standalone program from the command-line; and maybe only requiring an additional, previous compilation step, in which case, the compiler must also be excutable from the command-line. (*Student must use standard tools/compilers/etc., generally available for* **all** *popular platforms. Also, students* **must not** *submit source code that relies on or assumes that the resulting program will be run within a specific software-development IDE such as Microsoft's Visual Studio, Sun Microsystem's Eclipse, or Apple's Xcode.*)

**Collaboration Policy:** *While discussing general technical aspects of the project with peers is generally OK, each student must write and turn in their own report, code, and other required materials, based on the student's own work. It is OK to discuss the homework with peers but each student must write and turn in their own report, code, and other required materials, based on the student's own work.*